

IOS - IN-APP PURCHASE

http://www.tutorialspoint.com/ios/ios_in_app_purchase.htm

Copyright © tutorialspoint.com

In-App purchase is used to purchase additional content or upgrade features with respect to an application.

Steps Involved

Step 1. In iTunes connect, ensure that you have a **unique App ID** and when we create the application update with the **bundle ID** and code signing in Xcode with corresponding provisioning profile.

Step 2. Create a new application and update application information. You can know more about this in apple's [Add new apps](#) documentation.

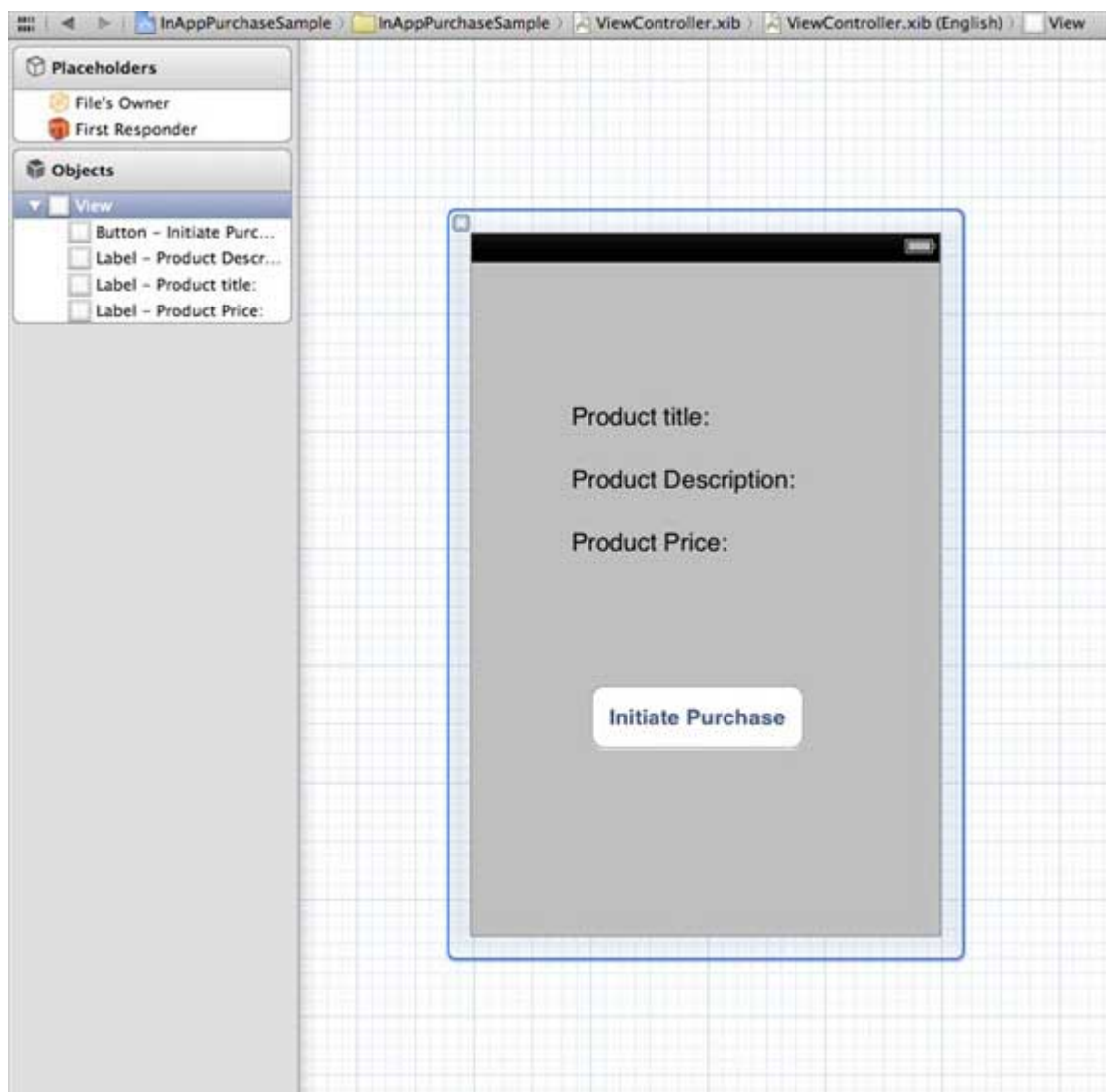
Step 3. Add a new product for in-app purchase in **Manage In-App Purchase** of your application's page.

Step 4. Ensure you setup the bank details for your application. This needs to be setup for **In-App purchase** to work. Also, create a test user account using **Manage Users** option in iTunes connect page of your app.

Step 5. The next steps are related to handling code and creating UI for our In-App purchase.

Step 6. Create a **single view application** and enter the bundle identifier is the identifier specified in iTunes connect.

Step 7. Update the **ViewController.xib** as shown below –



Step 8. Create **IBOutlet**s for the three labels and the button naming them as productTitleLabel, productDescriptionLabel, productPriceLabel and purchaseButton respectively.

Step 9. Select your project file, then select targets and then add **StoreKit.framework**.

Step 10. Update **ViewController.h** as follows –

```
#import <UIKit/UIKit.h>
#import <StoreKit/StoreKit.h>

@interface ViewController : UIViewController<
SKProductsRequestDelegate, SKPaymentTransactionObserver>
{
    SKProductsRequest *productsRequest;
    NSArray *validProducts;
    UIActivityIndicatorView *activityIndicatorView;
    IBOutlet UILabel *productTitleLabel;
    IBOutlet UILabel *productDescriptionLabel;
    IBOutlet UILabel *productPriceLabel;
    IBOutlet UIButton *purchaseButton;
}
- (void)fetchAvailableProducts;
- (BOOL)canMakePurchases;
- (void)purchaseMyProduct:(SKProduct*)product;
- (IBAction)purchase:(id)sender;

@end
```

Step 11. Update **ViewController.m** as follows –

```
#import "ViewController.h"
#define kTutorialPointProductID
@"com.tutorialPoints.testApp.testProduct"

@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Adding activity indicator
    activityIndicatorView = [[UIActivityIndicatorView alloc]
initWithActivityIndicatorStyle:UIActivityIndicatorViewStyleWhiteLarge];
    activityIndicatorView.center = self.view.center;
    [activityIndicatorView hidesWhenStopped];
    [self.view addSubview:activityIndicatorView];
    [activityIndicatorView startAnimating];
    //Hide purchase button initially
    purchaseButton.hidden = YES;
    [self fetchAvailableProducts];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (void)fetchAvailableProducts{
    NSSet *productIdentifiers = [NSSet
initWithObjects:kTutorialPointProductID, nil];
    productsRequest = [[SKProductsRequest alloc]
initWithProductIdentifiers:productIdentifiers];
    productsRequest.delegate = self;
    [productsRequest start];
}
```

```

}
- (BOOL)canMakePurchases
{
    return [SKPaymentQueue canMakePayments];
}
- (void)purchaseMyProduct:(SKProduct*)product{
    if ([self canMakePurchases]) {
        SKPayment *payment = [SKPayment paymentWithProduct:product];
        [[SKPaymentQueue defaultQueue] addTransactionObserver:self];
        [[SKPaymentQueue defaultQueue] addPayment:payment];
    }
    else{
        UIAlertView *alertView = [[UIAlertView alloc] initWithTitle:
@"Purchases are disabled in your device" message:nil delegate:
self cancelButtonTitle:@"Ok" otherButtonTitles: nil];
        [alertView show];
    }
}
- (IBAction)purchase:(id)sender{
    [self purchaseMyProduct:[validProducts objectAtIndex:0]];
    purchaseButton.enabled = NO;
}

#pragma mark StoreKit Delegate

- (void)paymentQueue:(SKPaymentQueue *)queue
updatedTransactions:(NSArray *)transactions {
    for (SKPaymentTransaction *transaction in transactions) {
        switch (transaction.transactionState) {
            case SKPaymentTransactionStatePurchasing:
                NSLog(@"Purchasing");
                break;
            case SKPaymentTransactionStatePurchased:
                if ([transaction.payment.productIdentifier
isEqualToString:kTutorialPointProductID]) {
                    NSLog(@"Purchased ");
                    UIAlertView *alertView = [[UIAlertView alloc] initWithTitle:
@"Purchase is completed succesfully" message:nil delegate:
self cancelButtonTitle:@"Ok" otherButtonTitles: nil];
                    [alertView show];
                }
                [[SKPaymentQueue defaultQueue] finishTransaction:transaction];
                break;
            case SKPaymentTransactionStateRestored:
                NSLog(@"Restored ");
                [[SKPaymentQueue defaultQueue] finishTransaction:transaction];
                break;
            case SKPaymentTransactionStateFailed:
                NSLog(@"Purchase failed ");
                break;
            default:
                break;
        }
    }
}

- (void)productsRequest:(SKProductsRequest *)request
didReceiveResponse:(SKProductsResponse *)response
{
    SKProduct *validProduct = nil;
    int count = [response.products count];
    if (count>0) {
        validProducts = response.products;
        validProduct = [response.products objectAtIndex:0];
        if ([validProduct.productIdentifier
isEqualToString:kTutorialPointProductID]) {
            [productTitleLabel setText:[NSString stringWithFormat:
@"Product Title: %@", validProduct.localizedTitle]];
        }
    }
}

```

```

        [productDescriptionLabel setText:[NSString stringWithFormat:
@"Product Desc: %@",validProduct.localizedDescription]];
        [productPriceLabel setText:[NSString stringWithFormat:
@"Product Price: %@",validProduct.price]];
    }
} else {
    UIAlertView *tmp = [[UIAlertView alloc]
initWithTitle:@"Not Available"
message:@"No products to purchase"
delegate:self
cancelButtonTitle:nil
otherButtonTitles:@"Ok", nil];

    [tmp show];
}
[activityIndicatorView stopAnimating];
purchaseButton.hidden = NO;
}

@end

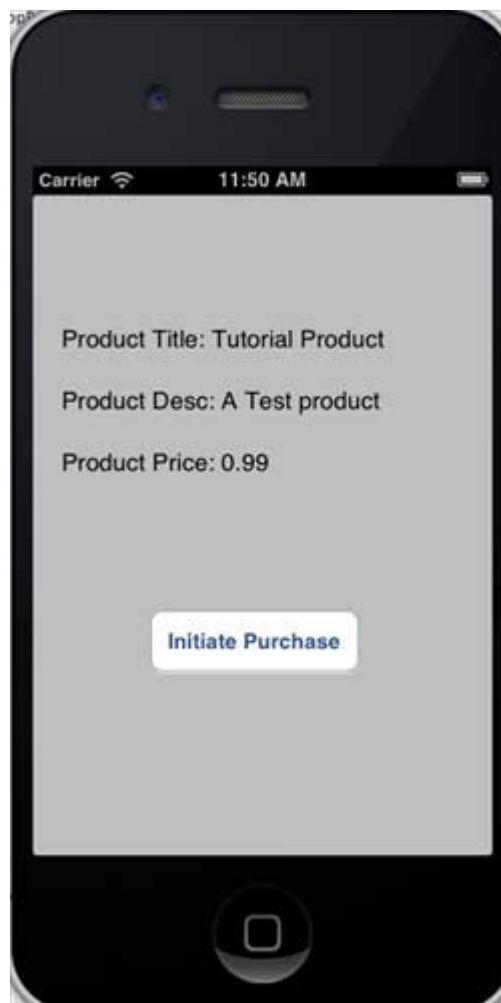
```

Note

You have to update kTutorialPointProductID to the productID you have created for your In-App Purchase. You can add more than one product by updating the productIdentifiers's NSSet in fetchAvailableProducts. Similarly, handle the purchase related actions for product IDs you add.

Output

When we run the application, we'll get the following output –



Ensure you had logged out of your account in the settings screen. On clicking the Initiate Purchase, select Use Existing Apple ID. Enter your valid test account username and password. You will be shown the following alert in a few seconds.



Once your product is purchased successfully, you will get the following alert. You can see relevant code for updating the application features where we show this alert.



